Impact of Neuron Models and Network Structure on Evolving Modular Robot Neural Network Controllers

Leo Cazenille
TAO/LRI
Univ. Paris-Sud, CNRS, INRIA
F-91405 Orsay, France
leo.cazenille@lri.fr

Nicolas Bredeche
TAO/LRI
Univ. Paris-Sud, CNRS, INRIA
F-91405 Orsay, France
nicolas.bredeche@lri.fr

Heiko Hamann
Art. Life Lab, Zoology,
Karl-Franzens University Graz
8010 Graz, Austria
heiko.hamann@uni-graz.at

Jürgen Stradner
Art. Life Lab, Zoology,
Karl-Franzens University Graz
8010 Graz, Austria
juergen.stradner@uni-graz.at

ABSTRACT

This paper investigates the properties required to evolve Artificial Neural Networks for distributed control in modular robotics, which typically involves non-linear dynamics and complex interactions in the sensori-motor space. We investigate the relation between macro-scale properties (such as modularity and regularity) and micro-scale properties in Neural Network controllers. We show how neurons capable of multiplicative-like arithmetic operations may increase the performance of controllers in several ways whenever challenging control problems with non-linear dynamics are involved. This paper provides evidence that performance and robustness of evolved controllers can be improved by a combination of carefully chosen micro- and macro-scale neural network properties.

Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: Artificial Intelligence—*Robotics*

General Terms

Experimentations, Algorithms, Reliability

Keywords

Evolutionary Robotics, Artificial Neural Networks, Evolutionary Algorithms, Modular Robotics

1. INTRODUCTION

Evolving Artificial Neural Networks (ANN) for robotic control offers many interesting properties: ANN are easy to implement, easy to compute on-board, robust to noise, expressive and evolvable. In the recent years, several methods have been proposed to evolve ANN, through direct or indirect encodings, and with various macro-scale properties (e.g. regularity, modularity and hierarchy), either hand-crafted or obtained through spontaneous evolution (see next section for references).

However, properties at the level of a single neuron have been largely overlooked and most models proposed so far are based on variations of the seminal McCulloch and Pitts summing unit neuron (e.g. weighted sum of inputs, with a sigmoidal activation function). However, this original model is but a very basic interpretation of what can be observed in nature. Evidence from biology shows many cases of non-linear interactions between sensory inputs, which are addressed by unit (single neuron or population of neurons) performing multiplicative operations [12]. Multiplicative models have been introduced in Machine Learning [16, 4], and their benefits in term of expressivity have been thouroughly studied [17].

This is particularly relevant with modular robotics where non-linear interactions in the feature space are an important part of the problem of designing distributed controllers. This paper is related to our on-going research on this topic, where control is decentralized among autonomous robotic units assembled together into a larger robot, as illustrated in figure 1. In this setup, each robotic unit communicates with its direct neighbors and cooperation must be achieved so that the whole organism can address a given task [13].

This paper explores the benefits of macro-scale properties on network structure (regularity and modularity) and micro-scale properties at the level of a single neuron (summing vs. multiplicative operations) with respect to an evolutionary robotics problem involving non-linear dynamics and complex interactions between autonomous units. We consider an existing benchmark that is designed to match the properties of distributed control in modular robotics and we investigate the impact of several properties on control performance and robustness to generalization. We show that while modularity and regularity are important features for neural networks controllers, as advocated in other works, properties at the level of the neuron can also lead to an

GECCO'12, July 7-11, 2012, Philadelphia, Pennsylvania, USA.

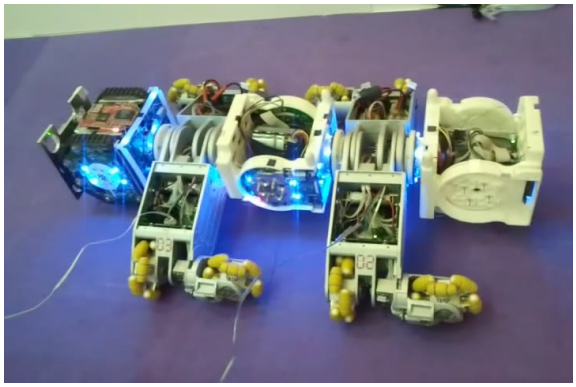


Figure 1: Five modules connected together to form a larger organism in the Symbrion project. Each module embeds its own controller and communicates with its neighbors.

increase in performance, generalization and speed of convergence, under certain conditions.

The paper is organized as follows: the next Section provides some background regarding both micro-scale properties and macro-scale Neural Networks properties. Then Section 3 describes the task and the Neural Networks models under scrutiny. Results and Analysis are presented in Section 4. Then, the last Section concludes and sketches future directions.

2. BACKGROUND

This section provides background information on micro-scale (neuron models) and macro-scale properties (network structures).

2.1 Neuron Models

In this review, we exclusively consider discrete time neural networks, due to the ease of implementation within on-board micro-controller with limited computational power and memory (e.g. spiking neurons offer original computational properties but also require additional computational resource which can be a problem when low-cost hardware is considered).

In this context, the classic summing unit is the most commonly used neuron model in Artificial Neural Networks. It is expressed as:

$$a_i = f(\sum_{j=1}^{N} w_{ji} * x_j + w_0), \quad (1)$$

where a_i is the activation of a given neuron, which results from the application of an activation function f on the weighted sum of signals from incoming neurons x_j (with $w_{ji} \in \mathbb{R}$), plus bias. Depending on the nature of f, this may corresponds to the classic threshold, sigmoidal or Radial Basis Function neuron models.

Most published works in Neuro-Evolution rely on this summing unit model as being the standard for neuron computation. Some few exceptions exists, in particular with respect to periodic functions, such as Karl Sims complex function for locomotion [18] or Compositional Pattern Producing Networks [19] (CPPN), which is not restricted to any particular kind of function.

While the original motivation for the summing unit model is based on its relevance to early biological data, as the original McCulloch and Pitts model suggests, other models have been explored, both in Biology and Computer Science. Multiplicative operations have been observed in various animals and experimental evidence has been found both at the level of population of neurons and single neurons [12].

In Computer Science, various multiplicative unit models have been introduced: the Sigma-Pi model [16], the Pi-Sigma model and the more general Product-Unit model (PUNN). The PUNN model [4] can be described as follows:

$$a_i = f(\prod_{j=1}^{N} x_j^{w_{ji}}) \quad (2)$$

Notations are similar as before, including $w_{ji} \in \mathbb{R}$, which is of utmost importance as negative exponents enable division operations. The number of exponents to multiply gives the *order* of the unit, hence the general term Higher-Order Neural Networks (HONN). Figure 2 shows a classical HONN topology where the hidden layer uses HONN units while the read-outs are summing units.

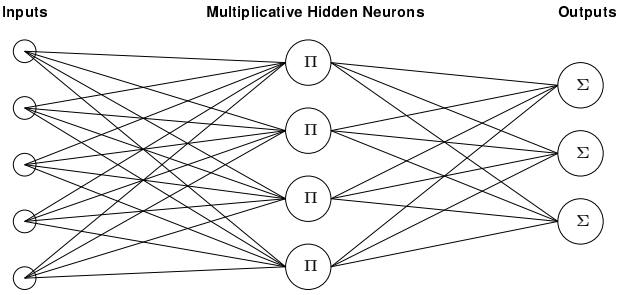


Figure 2: Example of Higher-Order Neural Network. The hidden layer embeds product units while the read-out neurons are summing units (possibly with sigmoid activation function).

Such models offer an increased computatioanal power as they can address non-linear interactions among sensory inputs. While multiplication and division can be accomplished using summing units, it would require much more neurons and layers just to match what a single multiplicative unit can do, in particular when real numbers and calculus precision are considered [17].

2.2 Network Structure

Several authors have highlighted the importance of topological structure for neuro-evolution [14, 9]. The main desired properties are modularity, regularity and hierarchy. These terms are defined as follow:

- Modularity: clearly identified localization of a specific element, either functional or structural;
- Regularity: repetitions of one or several patterns observed in the description;
- Hierarchy: recursive composition of a structure and/or function.

These properties have been studied both with respect to their functional relevance and possible spontaneous emer-

gence during the evolutionary process. For example, early works with Cellular Encoding implemented regularity through recursive decomposition, and was successfully applied to a six-legged robot locomotion problem [5]. Recent works focused on how such properties emerge during evolution, either with direct encoding [11, 2] or indirect encoding approaches [22]. The trade-off between capturing regularity and irregularities with indirect encoding methods has also been studied [3].

Another important property is related to feature space design, that is how sensory inputs are processed within the neural network. On the one hand, independent features with few interactions may be processed through different pathways and recombined in the last step (e.g. weighted aggregation). On the other hand, features may be strongly interacting and higher-level feature construction may be required to capture the information needed for the decision process. In Machine Learning, this is sometimes addressed as feature extraction [6], through feature decomposition (non-correlated features are treated separately) or construction (new features are constructed as combination of previous features). The challenge is to come up with a well-balanced trade-off between expressiveness and complexity (size of the feature space). As with other properties, feature space decomposition may or may not be relevant depending on the problem at hand. In the following, modularity is defined to enforce feature space decomposition (i.e. each module gets only part of the feature space as input).

3. METHOD

This section introduces the robotic task and representation formalisms considered to model the control architecture, each with specific micro- and macro-scale properties. Evolutionary setups and parameters are also presented.

3.1 The Robotic Task

The Coupled Inverted Pendulums benchmark introduced in [7] (and termed "multipole benchmark" in the following) extends the standard inverted pendulum scenario with non-linear dynamics. The goal of the multipole benchmark is to provide a challenge that is close to what one would expect from modular robotics scenarios, with interactions between autonomous parts and non-linear dynamics.

In this benchmark, pendulums are started in lower positions, hence, the non-linear upswinging phase is included, and the cart track length is restricted. In combination with a limited acceleration of the cart motor the upswinging cannot be managed by just moving back and forth once. In addition sampling rates of all sensors are limited, which is documented by the relation between the pre-defined cycle length τ of the controller and the maximal angular velocity of $0.05\pi[1/\tau] = 9°[1/\tau]$ (pendulum motion of up to 9° between two calls of the controller). The sensors do not deliver actual angles and positions directly, but instead partition the original values onto several intervals and sensors (a total of 10 sensors). The controllers have two outputs, left actuator A_0 and right actuator A_1 and the acceleration control of the cart is determined by their difference. Therefore, there is a total of 10 sensory inputs and 2 motor outputs. For each ANN model, there are two additional inputs and outputs to enable communication between controllers of connected, neighboring carts. These communication channels corresponds to additional inputs/outputs in the controller

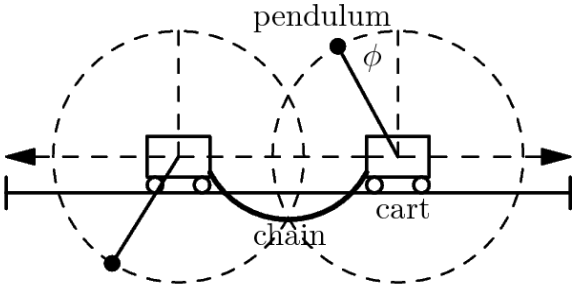


Figure 3: Coupled inverted pendulum benchmark with two carts, pendulums are free to move full 360° **mounted on the carts that move in one dimension (left/right) bounded by walls (track ends) and other carts. Marked angle is pendulum angle ϕ.**

for setups with more than one cart (the one-cart version has, of course, no communication).

In addition to the original definition of the benchmark, referred to as *multipole-velocity* from now, we consider an additional setup where the sensors for cart velocities and pendulum angular velocities are removed. This will be referred to as the *multipole-no-velocity* benchmark and it offers an increased challenge as velocities may be required, and thus reconstructed, to completely address the balancing problem. Due to the removal of velocity sensors, the multipole-no-velocity benchmark provides 6 (+2 for communication) sensory inputs instead of 10 (+2). An important fact is that when coupled carts are considered, it is theoretically possible to obtain memory-based behaviors even if the controllers are not capable of memorization as information can be transmitted back and forth between controllers of the carts.

The coupling of carts by chains is the most important difference to the standard inverted pendulum. Carts can move independently as long as they do not pull a chain or run into each other. Hence, each cart has to avoid other carts and walls (cart track ends) and has to balance its pendulum at the same time. In the following investigations, the cart number is increased without changing the track length so as to model problems of growing difficulty.

The fitness function is basically the percentage of time steps that all pendulums spent in the upper equilibrium position ($\phi = 0$) – the higher the fitness, the better the behavior. Deviations from $\phi = 0$ are linearly scaled, that is, $\phi = 0.5\pi$, for example, is evaluated as '50% in upper position'. If any constraint is violated (e.g., cart runs into other cart, cart runs into chain, cart runs into wall, pendulum velocity too high, etc.) an evaluation run is aborted and the fitness is reduced proportionally to the elapsed time. An extensive description can be found in [8].

3.2 The Controllers

Each pole is controlled by an Artificial Neural Networks, which can communicate with its direct neighbors. As with homogeneous distributed control in modular robotics, only one neural algorithm is evolved for control, which is considered as a template and duplicated within each cart. As a consequence, each cart will behave differently depending on its initial position, sensory inputs and past experience.

Controllers have been designed so that it can be possible

to evaluate the relevance of specific properties, both at the level of the network structure and neuron model. In particular, we aim to address the following questions: Is there a benefit in modularity in processing specific inputs through well identified group of neurons? Is there a benefit in regularity as evolving few patterns (i.e. groups of neurons with a given structure and weights), duplicated to form a final, larger, network? What is the benefit of using multiplicative neurons compared to summing units? The following models are devised to address these questions:

HONN and MLP: Higher-Order Neural Networks (HONN) and Multi-Layered Perceptron (MLP) are similarly defined as Multi-Layered Neural Networks: the input and output layers are fully connected to the hidden layer, and neurons for the output layer use a summing unit model with a sigmoidal activation function. They differ only in that MLP gets only one hidden layer of sigmoidal summing units while the HONN gets one hidden layer of multiplicative units.

C-MLP and C-HONN (i.e. with compartment): The notion of "compartment" (i.e. a structural module) refers to dedicated groups of neurons that are connected to only one input and one output. Figure 4 illustrates the structure of a Compartment Artificial Neural Networks. For C-MLP, a compartment is an MLP with one input, a hidden layer of sigmoid summing units, and one output with linear activation function. A C-HONN is similarly defined, except that there are two hidden layers, the first with multiplicative units, and the second with summing units, and a multiplicative unit as output (sigmoid activation functions are used for all neurons). In both cases, read-out neurons for the whole network are summing units with linear activation function. The number of compartments is fixed *a priori*, but the connection to one specific input and output is evolved, as well as the weights within each compartment. Weights connecting each compartment to its selected input and selected read-out neurons are also evolved. To some extent, this is an extreme case of feature space decomposition where the whole network may be considered as a mixture of experts, each expert specialized for processing a specific feature.

C-MLP-WP and C-HONN-WP (i.e. with pattern(s)): As previously, the neural network structure is organized into compartments with one input and one output. All compartments, however, may be identically defined as an instance of a reference evolved pattern. A pattern is defined as a compartment template, i.e. a neural network with one input, one output and one (MLP) or two (HONN) hidden layers. The motivation is to explore the impact of regularity in the network. In the extreme case, only one pattern may be evolved, and cloned during the actual construction of the neural network. As previously, weights connecting each compartment to its selected input and output are evolved.

In addition, we considered several well established neuroevolution methods from the literature: Echo State Networks [10] (ESN), NEAT [21] and HyperNEAT [20] (used to generate an ANN which is then used for control). It should be noted that the original HyperNeat implementation used for this experiment has been shown to produce sub-optimal solutions whenever regularity is required [3]. Hence, different results may be obtained with further extensions of this algorithm that are able to deal with regularity [22]. However, we highlight that the goal here is to provide a baseline with well-known algorithms from the literature, so as to advocate for the relevance (or not) of micro- and macro-scale properties in various experimental setups.

We also considered the Artificial Homeostatic Hormone Systems (AHHS), initially used for the multipole-velocity version of the benchmark, and which provides the best solution so far (compared to NEAT) on performance and generalization. To some extent, AHHS also models several properties evoked earlier: feature space is decomposed as each feature is processed separately by specialized "rules" and multiplicative operations can be computed through virtual "hormones", that are used to transmit information in the system. An extensive description can be found in [8].

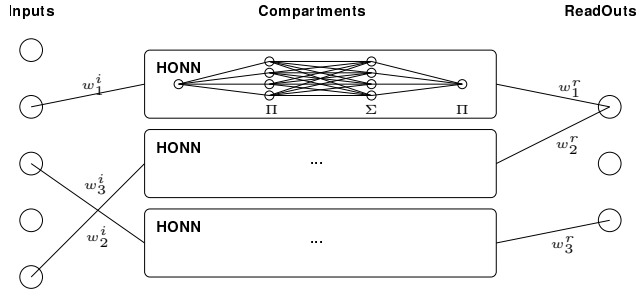


Figure 4: Structural description of a C-HONN. A C-MLP is described in a similar fashion by replacing the embedded HONN networks by MLP networks with one hidden layer. Similarly, C-HONN-WP, resp. C-MLP-WP, accept the same description, with embedded HONN, resp. MLP, as copies of a specific reference template (i.e. an evolved pattern).

3.3 Evolutionary Optimization

All models (except Neat, HyperNEAT and AHHS) are optimized using the state-of-the-art CMA-ES [1] algorithm. The official rtNeat and HyperNEAT-C++ implementations are used for the corresponding models. The original AHHS implementation is used [7].

A fixed budget of 20,000 (resp. 50,000) evaluations is set for the multipole-velocity (resp. multipole-no-velocity) benchmarks. 30 independent runs have been performed for each setup (NN model, cart number, with/without velocity information). In the next section, all claims consider statistically significant measure based on Wilcoxon tests (i.e. with p-value < 0.05).

A large number of preliminary experiments were performed to find operational experimental parameters for each approach. Table 5 summarizes parameters for each approach, as well as comments on robustness of parameters and genome size for each setup (except for Neat and HyperNEAT, which use variable-length genomes). Genome sizes vary to a large extent, depending on the structure of the controllers, but the evaluation budget remains the same to provide fair comparison. Note that the source code for running all the experiments is freely available[1].

4. RESULTS

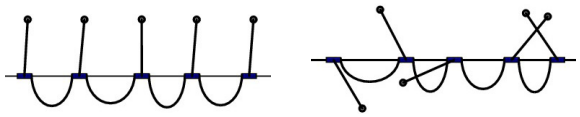
Figure 7 show the results from experiments on the multipole-velocity and multipole-no-velocity benchmarks, using a sin-

[1]http://pages.isir.upmc.fr/evorob_db/

Method	Parameters	Comments	Genome sizes
MLP	20 hidden neurons (with bias)		262, 344, 182, 264
NEAT	Parameters as in [23]		n/a
HyperNEAT	HyperNEAT-C++ default parameters, recurrence and self-recurrence enabled		n/a
ESN	reservoir size=100, density=0.5, damping=0.88	robust to density (values of 0.1, 0.2, 1.0 provide similar results)	200, 400, 200, 400
AHHS	Parameters as in [7]. 2 compartments per controller, 30 rules, 1 (multipole-velocity) to 3 (multipole-no-velocity) hormones		363, 369, 363, 369
C-MLP	30 compartments, each compartment is an MLP(1:20:1) (i.e. 1 input, 20 hidden neurons, 1 output)	bias is not mandatory	1320,1320,1320,1320
C-MLP-WP	30 compartments, each compartment is an MLP(1:20:1) (same as previous), 1 pattern	increasing the number of patterns up to 60 does not make any difference, bias is not mandatory	190, 190, 190, 190
HONN	multipole-velocity: 1 layer of 20 hidden multiplicative neurons ; multipole-no-velocity: 1 layer of 100 hidden multiplicative neurons, no bias	multipole-no-velocity requires larger networks	160, 240, 800, 1200
C-HONN	30 Compartments of HONN(1:4:4:1) (i.e. 1 input, 4 multiplicative units, 4 summing units, 1 multiplicative unit as output)	two hidden layers is mandatory	840, 840, 840, 840
C-HONN-WP	30 Compartments of HONN(1:4:4:1) (same as previous), 5 patterns	requires 5 (or more) patterns	270, 270, 270, 270

Figure 5: Parameters for all the models. The third column gives genome size for (a) multipole-velocity with 1 cart; (b) multipole-velocity with > 1 carts; (c) multipole-no-velocity with 1 cart; (d) multipole-no-velocity with > 1 carts

gle cart, and then 3 and 5 coupled carts. Boxplots aggregate the best results from each run obtained from each models and setups (i.e. a total of 1800 runs - i.e. 30 runs, 10 models, 6 setups). The one-cart versions of the benchmark roughly corresponds to the traditional inverted pendulum benchmark while the 3- and 5-carts versions feature both non-linear dynamics and complex interactions between carts.



(a) (b)

Figure 6: Snapshots of the best behaviors obtained for 5 carts in the (a) multipole-velocity and (b) multipole-no-velocity benchmarks.

Results on the multipole-velocity benchmark (see Figs. 7(a), 7(c), 7(e)) show that while well-established neuro-evolutionary algorithms (MLP, ESN, Neat, HyperNeat) perform extremely well on the simplest setup (1 cart), performance drops dramaticaly whenever more carts are considered. On the other hand, all other models (except for HONN) remain remarkably stable for any number of carts considered. For the more difficult multipole-velocity with 3 or 5 carts, models with compartments (incl. AHHS) dominate in term of average performance. It is even more true for pattern-enabled models (C-MLP-WP, C-HONN and C-HONN-WP) which provide the best results whenever more than one cart is involved.

Several observations can be drawn from these results: firstly, it appears that modularity (using compartments) and regularity (using duplicated patterns) are key features as difficulty grows. Secondly, multiplicative units (HONN) appears as a counter-productive property if used alone, while they yield to the best scores when combined with modularity (and regularity) on the more complex setups. Also, the good performance of decomposing the feature space, thanks to specialized compartments with one input each, may advocate that the multipole-velocity does not require complex interactions between inputs to be solved.

The fact that modularity and regularity are relevant properties is not a new idea and has been claimed elsewhere on many occasions (cf. section 2.2). However, it is interesting to note that both Neat and HyperNeat are unable to reach the best performance while they could theoretically evolve such properties in the controllers: this issue relates to the trade-off between expressivity and evolvability that have been addressed elsewhere (e.g. see [22] on enabling HyperNeat to generate regularity and modularity).

Let us now consider the multipole-no-velocity benchmark. Results with one cart are roughly the same as previously except for HONN controllers which go from the worst performing controllers (if velocity information is available) to the top performing controllers, implying that multiplicative units can be exploited in the current setup. Another surprising results is achieved with MLP, which successfully succeeds in balancing the pole. Looking carefully at the evolved strategies, we found that the one-cart multipole-no-velocity

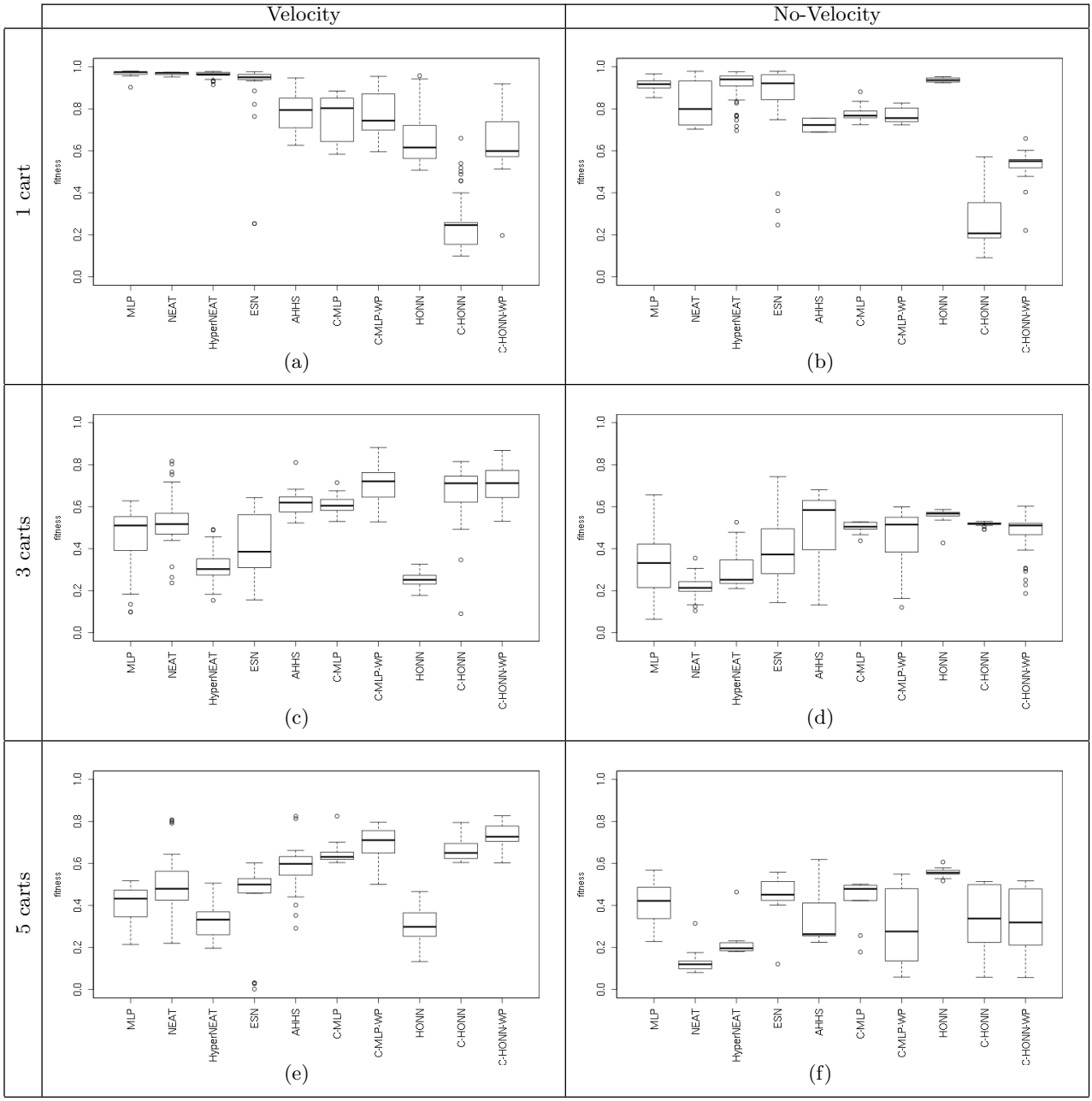


Figure 7: Performance of all controllers. For a given model and configuration, each boxplot aggregates the best individuals from 30 independant runs (i.e. 30 values).

can indeed be solved without memory capabilities: using the distance sensors to the wall, a simple MLP is able to learn a perfect trajectory from the cart initial position to the balanced position. While this trick is possible with one cart, the complex dynamics involved when several carts are considered did not permit to find such a solution in further experiments. At this point, an important fact to keep in mind is that a controller from a cart is able to communicate with its neighbors. As a consequence, memorizing information is possible even when the controller is a feed-forward neural network as information can be transmitted back and forth between carts.

Results are quite different when more than one cart is considered. HONN remains unchallenged, closely followed by AHHS, C-MLP, C-MLP-WP, C-HONN and C-HONN-WP when three carts are considered. On the 5-carts version, AHHS performance drops while C-MLP, ESN and MLP provide competitive results (still below those of HONN). The fact that MLP and C-MLP are still well ranked can be explained by looking at the resulting behaviors. Figure 6 illustrates the best behaviors for 5 carts obtained in both benchmarks. While the multipole-velocity benchmark is solved by most of the models with a median value > 0.6, no model achieves the optimal behavior whenever velocity information is removed. Instead, best behaviors correspond to swinging poles, accelerating or decelerating depending on the position

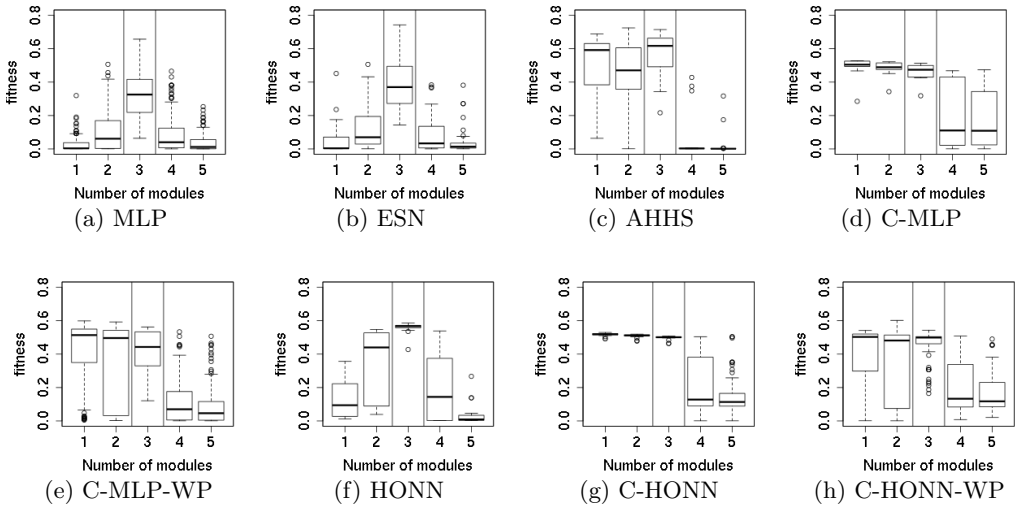


Figure 8: Generalization capability of the best controllers evolved for three carts, in the multipole-no-velocity benchmark. Each controller is re-evaluated with 1, 2, 4, and 5 carts to assert its transferability in a different setup. Neat and HyperNeat are omitted due to poor performance (similar to MLP and ESN).

of the pole with respect to the horizon so as to maximize the time spent near the equilibrium position while avoiding the other carts. In the 3-carts multipole-no-velocity, evolved behaviors are similar with some exceptions for some HONN and ESN controllers that achieve equilibrium for one cart, and adopt swinging behaviors for the two other carts.

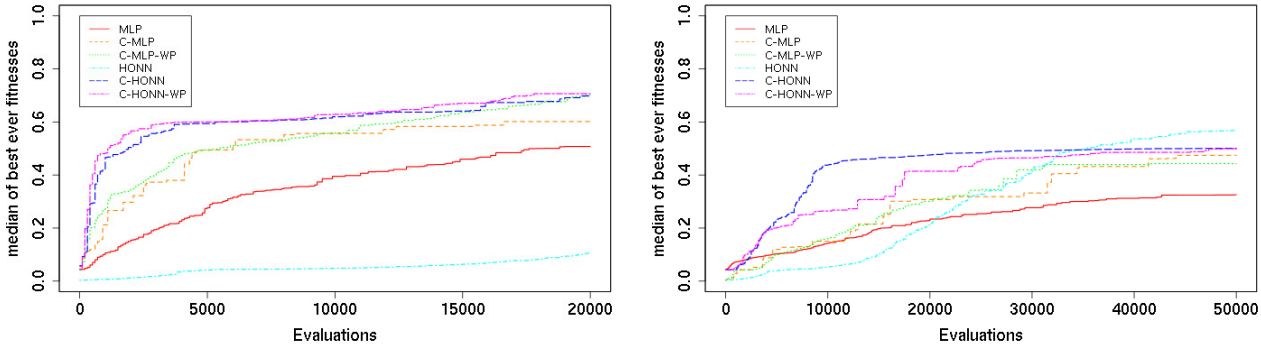
What makes a significant difference between these approaches is to be found in their generalization capabilities. The goal is to evaluate how controllers for each model evolved for the 3-carts version of the multipole-no-velocity perform when tested with 1, 2, 4 or 5 carts. Firstly, in the multipole-velocity benchmark, 3-carts evolved controllers from all approaches yield similar good results with different number of carts, except for MLP, ESN, and Neat which completely fail the generalization test. Secondly, the multipole-no-velocity provides an increased challenge with respect to generalization, as shown in figure 8. The supposedly nearly best results in performance with HONN (and, to some extent, with MLP) are strongly counter-balanced by their very bad results in generalization. Then, only C-MLP and C-HONN controllers (and to less extent C-MLP-WP and C-HONN-WP controllers) produce similar perfomance with 1 or 2 carts while still performing quite well with 4 or 5 carts, at least for some individuals (cf. upper quartiles, maximum values and outliers).

From the results in performance and generalization, it appears that both macro-scale properties and micro-scale properties are relevant to achieve good performance and generalization. In the multipole-velocity benchmark, using modularity (i.e. with compartments) or, even better, modularity combined with regularity (i.e. with patterns) provide the best results, and using multiplicative unit neurons even increases performance as C-HONN and C-HONN-WP both improve over the performance of their MLP equivalents. In the multipole-no-velocity, generalization plays a key role for comparing the different models: this time, modularity alone provides the best controllers on performance *and* generalization, and using summing units leads to slightly better results than using multiplicative units. However, a closer look at the actual course of the evolutionary process advocates for the benefits of using multiplicative units: figure 9 shows progress throughout the evolutionary process for the various flavors of additive- and multiplicative-based models and establish that C-HONN and C-HONN-WP controllers improves *much* faster than any other methods on the most complex versions of the multipole-no-velocity benchmark, especially with respect to other models capable of generalization (results are similar for both the 3 and 5 carts configurations). As a matter of fact, C-HONN almost converged in less than 10,000 evaluations (followed by C-HONN-WP), while other methods require more than 40,000 evaluations to reach similar performance.

5. CONCLUSIONS

This work demonstrates the importance of micro-scale properties in Artificial Neural Networks, in particular with control problems that are characterized by non-linear dynamics and complex interactions between autonomous systems, as it is often the case in modular robotics. The benefits of neuron models that enable multiplicative operations have been demonstrated in combination with other macro-scales properties, in particular when modularity is involved. The take-home message advocates for the use of multiplicative units in neuro-evolutionary algorithms: multiplicative units used in conjunction with modularity, or modularity with regularity, always lead to at least similar or better results when complex dynamics are involved, in much less evaluations.

This work assumed a straight-forward approach to implementing modularity and regularity as explicit features of neural network controllers. This was justified as the goal was to evaluate the positive or negative impacts of such properties. However, and with respect to further use, relaxing this assumption may lead to more adaptive neuro-evolutionary algorithms that can exploit results obtained here. For example, recent works on spontaneous evolution of such network properties [2, 22] suggest that the decision to use summing and/or multiplicative units may be evolved as well, to match the requirements of the problem at hand.



(a) multipole-velocity, 3 carts

(b) multipole-no-velocity, 3 carts

Figure 9: Evolutionary runs: tracking improvements throughout evaluations using medians of the best-ever individuals from each runs. (a) multipole-velocity, 3 carts ; (b) multipole-no-velocity, 3 carts. Note that results with 5 carts are similar (not shown due to page limit restriction).

Hierarchy as a macro-scale property has not been discussed in this paper while it could be combined with the properties evoked earlier. Indeed, preliminary experiments yielded limited results, which are mostly due to the difficulty of defining an explicit encoding allowing for hierarchical organization of compartments. This also advocates for an indirect encoding representation, that is leaving the algorithm to decide on the structure of the controllers, rather than enforcing it. To this end, extensions of this work towards integration of evolving micro-scale properties in Cartesian GP [15] and/or HyperNeat-like methods are currently under investigation.

We conclude this paper with a last remark on the multipole benchmarks. The no-velocity version remains very challenging for all approaches as only sub-optimal solutions have been found whenever there is more than one cart. Moreover, it offers an interesting environment with regards to generalization because some methods appear to be much more efficient than others with respect to the transferability of the controllers. Also the memory issue is addressed because operations requiring memorization (such as reconstruction of velocity information) may be achieved through sharing information between several carts.

Acknowledgments

This work was made possible by the European Union FET Proactive Initiative: Pervasive Adaptation funding the Symbrion project under grant agreement 216342. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr). We also would like to thank Jean-Marc Montanier and Taras Kowaliw for many helpful scientific and technical discussions.

References

[1] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *Proc. CEC 2005*, pages 1769–1776. IEEE Press, 2005.

[2] J. C. Bongard. Spontaneous evolution of structural modularity in robot neural network controllers hardware. In *GECCO*, pages 251–258, 2011.

[3] J. Clune, K. Stanley, R. Pennock, and C. Ofria. On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Ev. Comp*, 2011.

[4] R. Durbin and D. Rumelhart. Product units: A computationally powerful and biologically plausible extension to backpropagation networks. *Neural Computation*, 1(1):133–142, 1989.

[5] F. Gruau. Modular genetic neural networks for six-legged locomotion. In *Artificial Evolution*, 1995.

[6] I. Guyon. *Feature extraction: foundations and applications*, volume 207. Springer Verlag, 2006.

[7] H. Hamann, T. Schmickl, and K. Crailsheim. Coupled inverted pendulums: a benchmark for evolving decentral controllers in modular robotics. In *GECCO*, pages 195–202, 2011.

[8] H. Hamann, T. Schmickl, and K. Crailsheim. A hormone-based controller for evaluation-minimal evolution in decentrally controlled systems. *Artificial Life*, 18(2), 2012.

[9] G. Hornby. Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1729–1736. AcM, 2005.

[10] H. Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the echo state network approach. Technical report, Fraunhofer Institute AIS, 2002. GMD Report 159.

[11] N. Kashtan and U. Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13773, 2005.

[12] C. Koch. *Biophysics of computation: information processing in single neurons*. Oxford University Press, USA, 2005.

[13] P. Levi and S. Kernbach, editors. *Symbiotic Multi-Robot Organisms*, volume 7 of *Cognitive Systems Monographs*. Springer Berlin Heidelberg, 2010.

[14] H. Lipson. Principles of modularity, regularity, and hierarchy for scalable systems. *Journal of Biological Physics and Chemistry*, 7(4):125, 2007.

[15] J. F. Miller and P. Thomson. Cartesian genetic programming. In *Proceedings of the European Conference on Genetic Programming*, pages 121–132, London, UK, 2000. Springer-Verlag.

[16] D. E. Rumelhart, G. E. Hinton, and J. L. McClelland. *A general framework for parallel distributed processing*, pages 45–76. MIT Press, Cambridge, MA, USA, 1986.

[17] M. Schmitt. On the complexity of computing and learning with multiplicative neural networks. *Neural Computation*, 14(2):241–301, 2002.

[18] K. Sims. Evolving virtual creatures. In *SIGGRAPH'94*, pages 15–22. ACM Press, July 1994.

[19] K. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, 2007.

[20] K. Stanley, D. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.

[21] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[22] P. Verbancsics and K. Stanley. Constraining connectivity to encourage modularity in hyperneat. In *GECCO*, pages 1483–1490, 2011.

[23] S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *The Journal of Machine Learning Research*, 7:877–917, 2006.